P2

Project report: the apple chess program

The Objective

I have been chosen to create a program of a well-known chess game, the ***Apple Chess***, also known as the ***Othello*** game. According to the instructions I got from my teacher, I need to create an Othello game that:

a.) Accepts two human players at a time,
b.) Design a proper user interface,
c.) Obeys the official rules of the game.

The Analysis

To accomplish the objectives I have mentioned, I must first learn to play the Othello game. From a several Internet web pages, I obtained the rules of the game. The following is a list of the rules.

1.) Black always plays first.
2.) The Othello chessboard must be an 8x8 square board.
3.) Flipping goes in eight different directions: horizontal, vertical, and diagonal. When a disc is placed in a certain position, it flips all of the opponent's disc in all direction until it reaches it's another disc of its own colour.
4.) Flipping of a disc must result in direct effect of a move, that means the flipped disc in one move will not flip the other discs around it.
5.) A legal move must satisfy two conditions:
i.) The move is on a position where no disc has been put on
ii.) The move can flip at least one of the opponent's discs
6.) Every move must be a legal move

7.) If no legal move is available for a player, that player must pass the game to the other opponent.
8.) Game ends in either condition:
i.) If no legal move is available for both players, it is a draw game.
ii.) All positions are occupied.

It is clear that all of the above rules but be carefully implemented into my program to ensure a good user experience.

The Design

The design of the program is simple, as I have mentioned earlier, an Othello game uses an 8 by 8 chessboard, to the users convenience, there must be some kind of an index system. After referring to some common chessboards, I decided that the interface should look like this:

	A	B	C	D	E	F	G	H
1								
2								
3				*				
4				O	X			
5				X	O			
6								
7								
8								

where X denotes black discs and O denotes white.

Using this system, the program will prompt the user for moves using this index. For instance, if the user would like to place a disc on the * position, the user can enter D3.

Implementation

Choosing the Right Language

To realize the design, I must choose a media and a programming language that is suitable for this project. Here is the list of some possible solutions.

Programming Language	Media	Software	Platform
C	Console	Any text editor	Any platform that has a C/C++ complier
C++	Console	Any text editor	Any platform that has a C++ complier
	GUI	Any text editor / Visual C++	Win16/Win32
Basic	GUI	Visual Basic	Win16/Win32
Perl	Console	Any text editor	Any platform that has a Perl interpreter.

To decide which programming language and media to use, I have thought of the benefits and drawbacks on using those mentioned above.

Programming Language		Advantages	Disadvantages
C		Extremely portable	Comes with fewer functions than C++
C++	Console	Extremely portable	Takes longer time to master
	GUI	Enhance user Experience	Takes even longer time to master / poor portability
Visual Basic		Easy to program	Poor portability
Perl		Easy to program	Execution time is longer because perl programs runs through interpreters

At first, I was going to make this program using Visual C++. However, because I don't know much about C++ and designing programs with a GUI under the windows environment, so I gave up this thought and turn to Perl, but Perl programs are not fast enough because it is interpreted by its interpreter, instead of compiling.

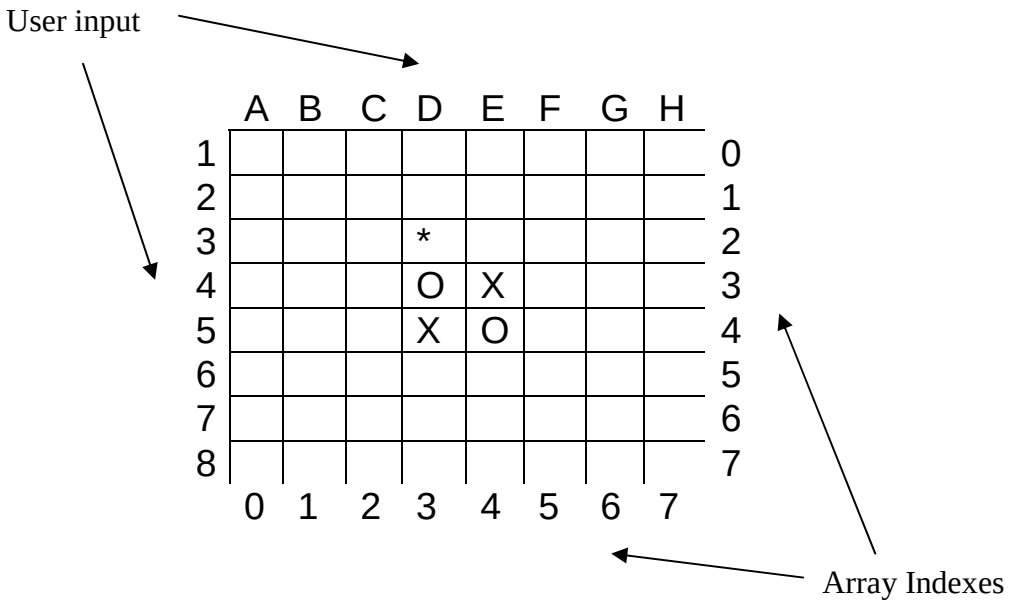
So, at the end, I chose to program in C because it takes less time to learn, it has a tremendous amount of documents on the Internet, and it will allow the program to become extremely portable.

The development platform

At first, I tried to work with Dev-C++ on windows, but it often generates error messages and failed to compile because some include files were missing. As my teacher provided me with another option, that is Linux, I eventually developed my program on it, using its text editors and C compliers which come with most distributions. I found that it was quite a good platform to work on because it can be accessed through the internet with ssh or telnet; it is free and robust; and it has many documentations on how things could be used.

Storing the data

It is obvious that the right place to store the chessboard is a two-dimensional array. Since array indexes in C starts from 0 (as illustrated below), I need to convert the incoming user input to what C understands. For instance, when the user inputs **D3**, I need to tell the program that it means **2,3** in the array.



So I wrote the function alpha2num() in the program to convert the column indexes back into the array index. Of course, I would have to initialize the array first before I could do anything.

The Flipping

At first glance, the flipping business may seem to be simple, but it turns out that it is somewhat complicated.
The program will check for all 8 directions, each direction has its own function, these functions will check:

1.) Check if in this direction there are opponent's discs,
2.) If so, it will continue until it reaches another disc of its own colour,
3.) If it reaches a blank position, it will stop checking and it will not flip any of those discs.
4.) If it exceeds the border of the chessboard, it will stop checking and will not flip any of those discs.
5.) If the next disc is of its own colour, it will not flip anything at all.

These functions took me a long period of time to debug and make them robust.

User Input & Error Handling

To prevent users from getting program errors with their input, it is necessary to create a mechanism to avoid such error from user input. This can be achieved by checking if the moves they entered are within the range of the chess board array and if the moves are valid. So every time they enter a move, the program will check if the desired position is available, once validated, it will write the move to the chess board array.

When does the game end?

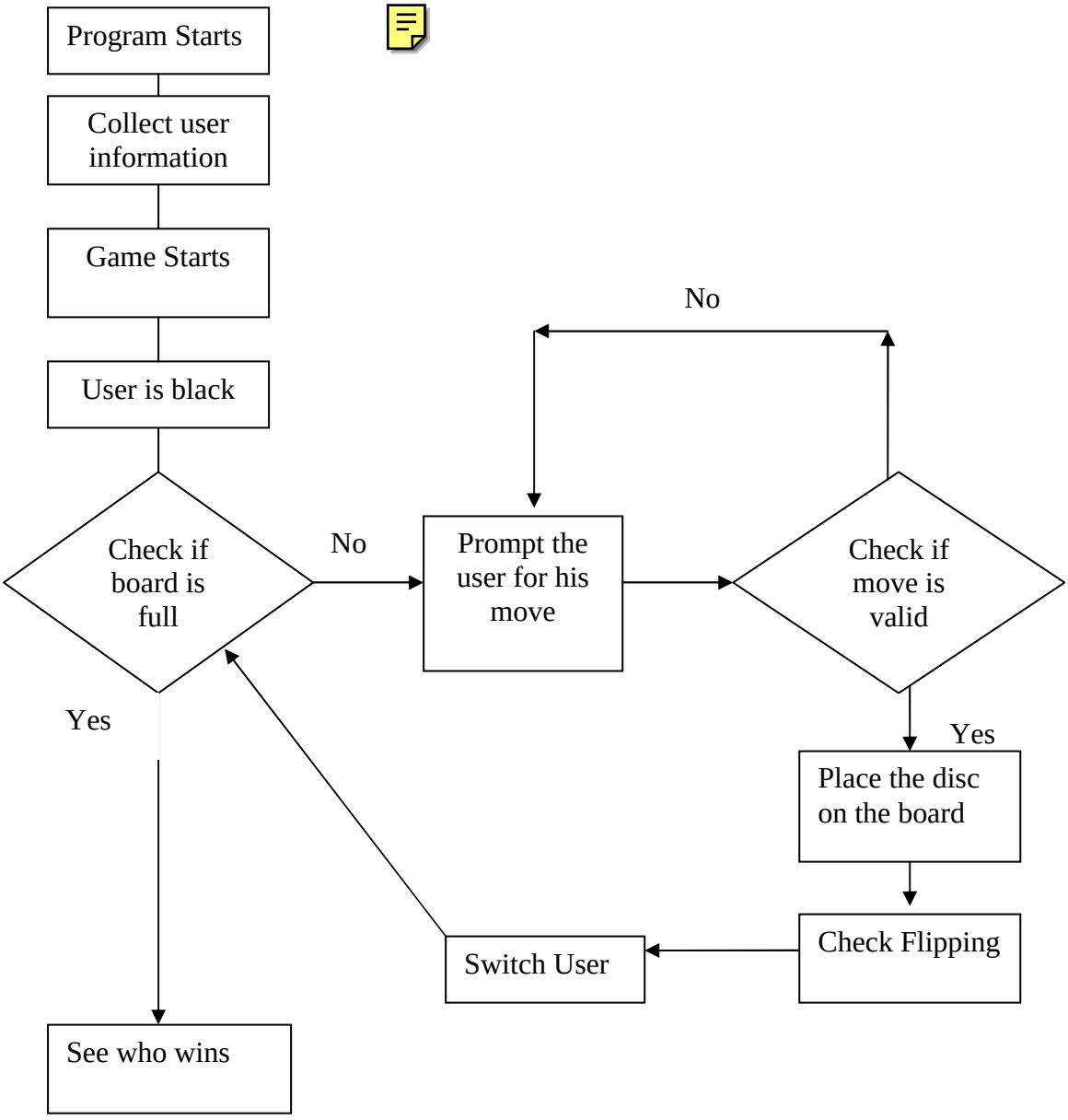
As stated earlier, the game ends in either condition:

i.) If no legal move is available for both players, it is a draw game.
ii.) All positions are occupied.

So I wrote a function to check every single position of the chessboard before the user input, to see if there are any legal moves for both players, if there is none, the game ends. If there are no legal moves available for the current player, it will display a message and tell the current player that he has no moves.

The program flow

Here is the actual program flow:



P2

User Guide

Requirements:
386 PC or Compatible
Windows 3.1/95/98/ME/2000/XP or Linux
(and many other platforms, provided that they have a C compiler)

Compilation:
Linux
In Shell, execute the following command:
gcc othello.c –o othello
(if gcc is not installed, please read the linux distribution's manual.)

Windows
Obtain any C/C++ compiler from the internet, and follow the instructions.

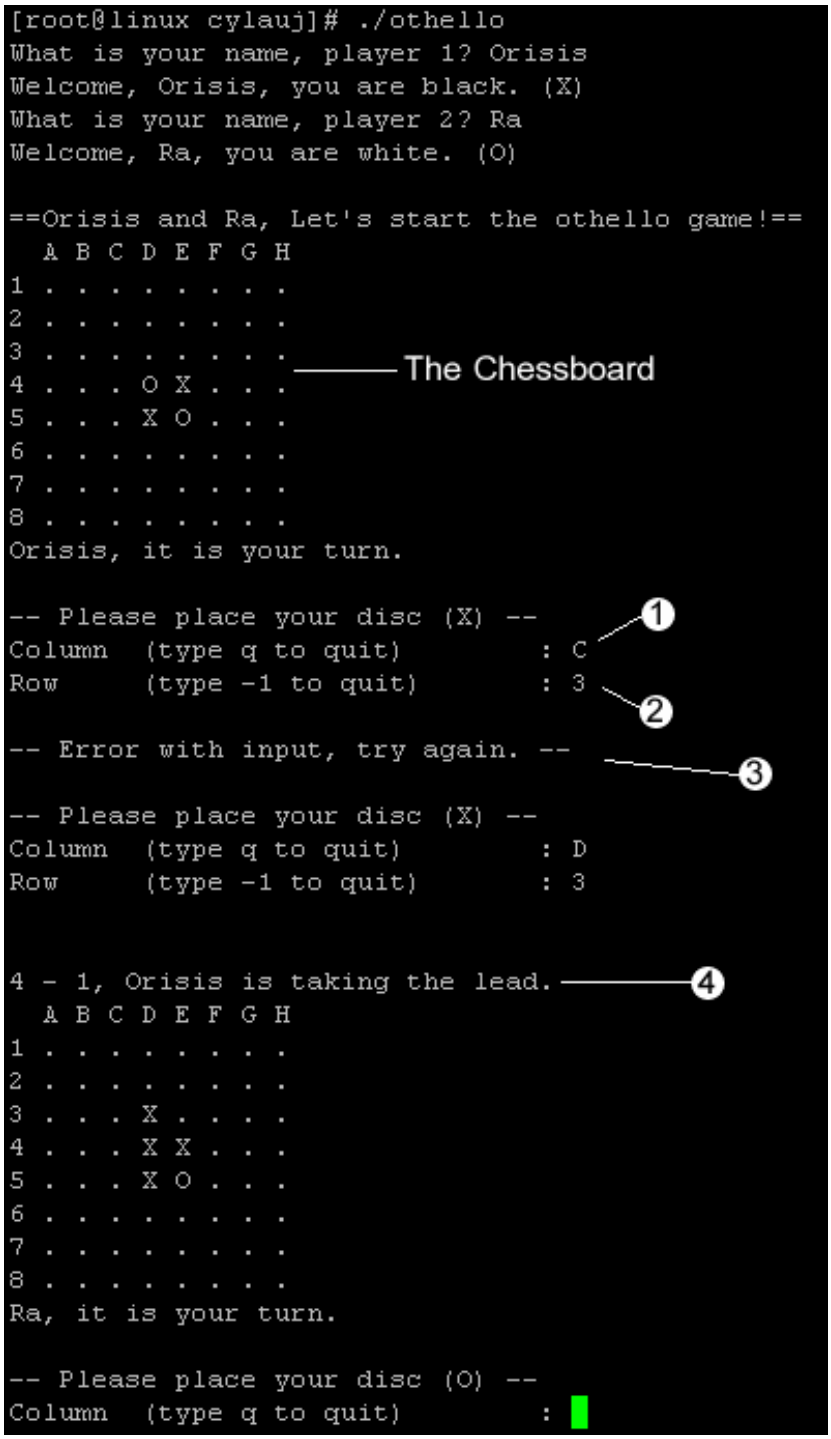
Installation:
No installation is required.

Running the program:
Linux
In the directory that stores your copy of the program,
type in the following command:
./othello

Windows
In MS-DOS, MS-DOS mode (Windows 95/98/ME), or command prompt (Windows 2000/XP).
Type :
C:\theDirectoryYouStoredIt>othello.exe

Playing the game

Please enter two names at the beginning of the game.
Player 1 will take the black discs, player 2 will take the white ones.



1&2	Where you input your move
3	This indicates that your move is not valid, you must place your disc where it can flip the discs of your opponent.
4	This displays who is taking the lead

To quit, press q when you are prompted to enter the column index or press -1 when you are prompted to enter the row index.

Testing and Evaluation

This program has been successfully compiled under both win32 environment and Linux environment. And some of the important features of this program have also been tested, as you will see below.

Pass

This is a situation where player 1 doesn't have any valid move and he needs to pass this game to his opponent, player 2.

```
14 - 12, 1 is taking the lead.
  A B C D E F G H
1 O O O O O O O .
2 O O O O X . . .
3 O X X X . . . .
4 . . X X X . . .
5 . . . X X . . .
6 . . . . X X X .
7 . . . . . . X .
8 . . . . . . . X
1, it is your turn.
Valid Moves for X : Sorry, you can't move! Press any key to pass.

  A B C D E F G H
1 O O O O O O O .
2 O O O O X . . .
3 O X X X . . . .
4 . . X X X . . .
5 . . . X X . . .
6 . . . . X X X .
7 . . . . . . X .
8 . . . . . . . X
2, it is your turn.
Valid Moves for O : 1,5 2,4 2,5 3,0 3,1 4,1 4,2 4,5 5,3 6,5
-- Please place your disc (O) --
Column  (type q to quit)        :
```

As you can see, the program successfully passes the game when a player runs out of moves.

Draw

This is a situation where both players possess the same number of discs at the end of the game.

```
  A B C D E F G H
1 O O O O X X X X
2 O O O O X X X X
3 O O O O X X X X
4 O O O O X X X X
5 O O O O X X X X
6 O O O O X X X X
7 O O O O X X X X
8 O O O O X X X X
Orisis, it is your turn.
This board is frozen!

=============== Result ===============
Orisis, you have 32 of (X) on the board.
Ra, you have 32 of (O) on the board.
Oops! Sorry, this is a draw game!
```

Since real draw games seldom appears in the game of Othello, I had modified the program to initialize half of the array as white discs and the other half as black discs, without modifying the core of the program. So this proves that the program will recognize a real draw game when the time comes.

Win

There is always a winner in a game. So after each game ends, the program must determine which player is the winner. It would count both players' discs on the chess board to see which of them got most discs.

```
21 - 43, 2 is taking the lead.
  A B C D E F G H
1 O O O O O O O X
2 O O O O O O X O
3 O O O O O O X O
4 O X O O O O O O
5 O O O O O O O O
6 O O O O O X X X
7 O O X X X X X X
8 X X X X X X X X

=============== Result ===============
1, you have 21 of (X) on the board.
2, you have 43 of (O) on the board.
Congratulations, 2, you won!
```

As seen from above, the program correctly displayed the number of the discs each player possesses and who the winner is. This time, it's player 2 who had won.

Conclusion & Discussion

After word

It is some kind of a challenge to write this program. To be honest, I have underestimated the complexity of this program. At first, I thought the logic and the nature of the othello game is simple enough to be converted into a computer program without a fuss. But, of course, it turned out that I was wrong.

During the process of creating this program, I have acquired some useful skills. I have learnt to compile programs written in C under both win32 and Linux environment. I have learnt to use vi, the very useful and popular text editor in the Linux world. And because my teacher had provided me a Linux environment to write this program, I learnt to use ssh to connect to other computers and transfer files though it. These are the skills that can be useful in many situations which I may face in the future. I am glad that I have this opportunity, it was an incredible experience.

Discussion

After all, I admit that this program is a very immature one. It has no A.I. capabilities, no network capabilities, not much of a friendly user interface. Here are some features that I could have done to enhance user experience and improves the program in general.

A 1-player game

After all, it is quite pointless for two people to sit behind the same computer to play one chess game. When there is no one around that you can play the game with, people wouldn't prefer to play with themselves since this chess game is sometimes an intellectual battle between two minds.
So the simple solution is to create a virtual opponent for the user. We could have had the computer to follow a certain strategies, so that it would appear to the player that the virtual opponent is a human player, flesh and blood.
Of course we could have let the player to decide the difficulty of the game, let them control how tough the virtual opponent would be. We could also let them to undo there moves or the computer's moves, to let them learn from there mistakes. But as I am not really a Othello master, I cannot set strategies and rules of engagement for the program to follow.

An Online game

As the internet has become a world full of such diverse culture and entertainment, why not add this game to the list? We could have allowed users to connect to other players using IP addresses. We could even have written a server program to accept a number of players as a directory service, to provide a place for users to meet other players. But as I am not quite familiar with networking in programming, I chose not to write such code.

A GUI version of the game

A program with a graphical interface can greatly enhance user experience. But since codes with GUI are extremely not portable, and I don't have much experience with GUI coding, I chose not to write a GUI for this program.